

# **Requirements Engineering 3: “Systems Engineering & Requirements Engineering”**

**Steve Easterbrook**

**9/3/97**

## **Outline**

**Application and Machine Domains**

**Phenomenology**

**Modeling**

**Classifying problem domains**

**Intro to systems engineering**

## Some definitions

- **‘Machine’**
  - We are interested in software systems
  - We will call the software system to be developed ‘the machine’
  - The hardware only exists to run the software, hence is also part of the machine
- **‘Application Domain’**
  - A machine will interact with its environment
  - A machine is built to serve some purpose in the world
  - The aspect of the environment that defines the machine’s purpose is its application domain
  - The application domain is often a human activity system.

Source: Adapted from Jackson, 1995, p72

3

## What vs. How

- **“Requirements should specify ‘what’ without specifying ‘how’”**
  - What does a web browser do?
  - What does a car do?
- **‘What’ refers to a system’s purpose**
  - it is external to the system
  - it is a property of the *application domain*
- **‘How’ refers to a system’s structure and behavior**
  - it is internal to the system
  - it is a property of the *machine domain*

Source: Adapted from Jackson, 1995, p207

4

## Application Domains

- **A machine is designed to be installed in the world to achieve some purpose**
  - Application domain is the aspect of the world that defines that purpose
- **Application domain  $\neq$  Environment**
  - (except when environment is defined solely in terms of the human activity system in which the machine is embedded)
- **Requirements only exist in the application domain**
- **Distinguishing between the machine and the application domain is the first and most crucial step in requirements engineering**

Source: Adapted from Jackson, 1995, p9-11

5

## Some examples

- **Airline Reservation System**
- **A payroll Accounting System**
- **A Flight Control System**
- **An Operating System**
- **A Print Utility**
- **A Web Browser**
- ***Your example here***

6

## Implementation Bias

- **Implementation bias is the inclusion of requirements that have no basis in the application domain**
  - i.e. mixing some 'how' into the requirements
- **Examples:**
  - The dictionary shall be stored in a hash table
  - The patient records shall be stored in a database

Source: Adapted from Jackson, 1995, p98

7

## Which are valid requirements?

- **The software shall be written in FORTRAN.**
- **The software shall respond to all requests within 5 seconds.**
- **The software shall be composed of the following 23 modules ....**
- **The software shall use the following fifteen menu screens whenever it is communicating with the user....**

Source: Adapted from Davis, 1990, p34

8

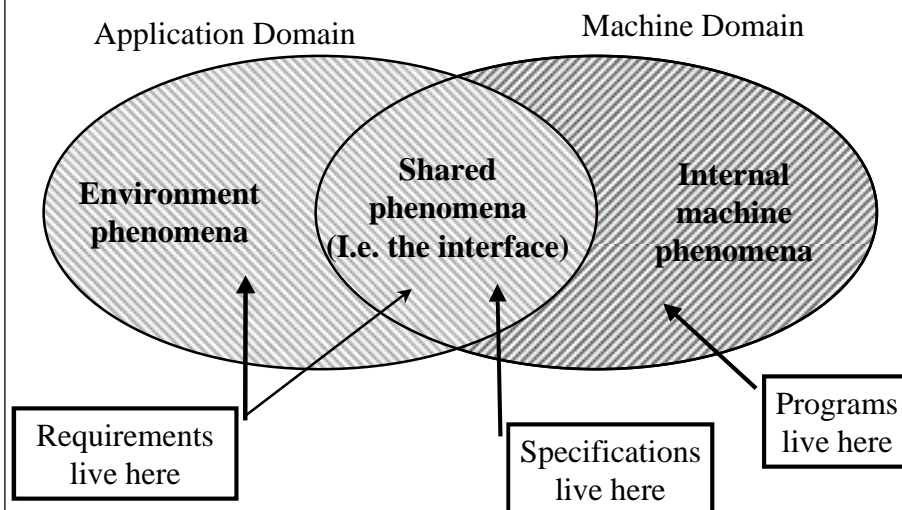
## Phenomena

- **Phenomena are what appear to exist when you observe the world**
  - phenomenology: the study of phenomena
  - ontology: the study of what really does exist (independently from any observer)
  - Weltanschauung: a world view that defines the set of phenomena that an observer is willing (likely) to observe ('viewpoint')
- **Any method embodies a particular viewpoint:**
  - OO sees the world as objects with internal state that respond to stimuli
  - SA sees the world as processes that transform data
  - Natural language also defines a viewpoint
- **By restricting the set of phenomena you can describe, a method restricts what you will observe**

Source: Adapted from Jackson, 1995, p143

9

## Shared Phenomenology

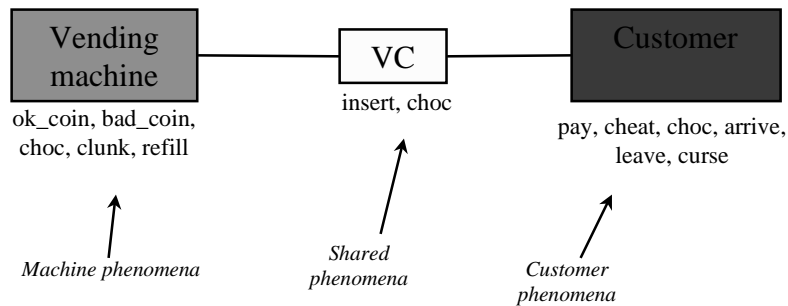


Source: Adapted from Jackson, 1995, p170

10

## Shared Phenomena

- E.g. vending machine:



Note: okay\_coin and bad\_coin both map to the insert event;  
pay and cheat both map to the insert event

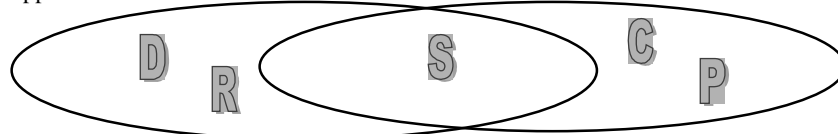
Source: Adapted from Jackson, 1995, p179-180

11

## Requirements as Application Phenomena

Application Domain

Machine Domain



- For a program to satisfy a requirement, we need to worry about:
  - The properties of the computer (C)
  - The properties of the program (P)
  - The properties of the machine in the application domain (I.e. the specification, S)
  - The properties of the domain independent of the machine (D)
  - The requirements for the machine (R)
- Demonstration that P satisfies R is then a two step process:
  - Do C and P imply S?
  - Do S and D imply R?

Source: Adapted from Jackson, 1995, p170-171

12

## Example

- **Requirement R:**
  - “Reverse thrust shall only be enabled when the aircraft is moving on the runway”
- **Domain Properties D:**
  - Wheel pulses on if and only if wheels turning
  - Wheels turning if and only if moving on runway
- **Specification S:**
  - Reverse thrust enabled if and only if wheel pulses on
- **S + D imply R**
  - But what if the domain model is wrong?

Source: Adapted from Jackson, 1995, p172

13

## In the mood

- **Mood (of a verb):**
  - Indicative: asserts a fact (“you sing”)
  - Interrogative: asks a question (“are you singing”)
  - Imperative: conveys a command (“Sing!”)
  - Subjunctive: states a possibility (“I might sing”)
  - Optative: expresses a wish (“may you sing”)
- **‘Shall’ and ‘will’ can be used in different moods:**
  - “I shall drown. No one will save me”
  - “I will drown. No one shall save me”
- **For requirements engineering:**
  - use the indicative mood for domain properties
  - use the optative mood for requirements
- **Never mix moods in the same description.**
  - Label the entire description with a single mood
  - Forget about linguistic subtleties within the description
  - Anyway, mood changes as development progresses!

Source: Adapted from Jackson, 1995, p125-127

14

## Indicative or Optative?

- The elevator never goes from the  $n$ th to the  $n+2$ th floor without passing the  $n+1$ th floor
- The elevator never passes a floor for which the floor selection light inside the car is illuminated without stopping at that floor
- If the motor polarity is set to *up*, and the motor switch setting changed from off to on the elevator starts to rise within 250ms
- If the *up* arrow indicator at a floor is not illuminated when the lift stops at the floor, it will not leave in an *upwards* direction
- The doors are never open at a floor unless the elevator is stationary at that floor
- When the elevator arrives at a floor, the *elevator-present* sensor at the floor is set to on.
- If an *up* call button at a floor is pressed when the corresponding light is off, the light comes on, and remains on until the call is serviced by the elevator stopping at that floor and leaving in an *upwards* direction.

Source: Adapted from Jackson, 1995, p126

15

## Fun with natural language...

- |                                                           |                                                                                 |
|-----------------------------------------------------------|---------------------------------------------------------------------------------|
| • Shake well before enjoying                              | • If party A calls party B and party B is idle, then party B's phone shall ring |
| • Shake well before opening                               |                                                                                 |
| • Shirts must be worn                                     | • No smoking                                                                    |
| • Your call will be answered in the order it was received | • Your satisfaction is guaranteed                                               |
|                                                           | • Maintain speed                                                                |

16

## Descriptions

- **A designation**
  - singles out a phenomena of interest
  - tells you how to recognize it
  - gives it a name
  - A designation is always informal, as it maps from the fuzzy phenomena to formal language
- **A Definition**
  - gives a formal definition of a term that may be used in other descriptions
  - Note: definitions can be more or less useful, but never right or wrong.
- **A refutable description**
  - states some property of a domain that could in principle be refuted
  - Might not be practical to refute it, but refutation should be conceivable
  - Refutability depends on an appeal to the designated phenomena of the domain being described
- **A rough sketch**
  - is a tentative description that is being developed
  - May contain undefined terms

Source: Adapted from Jackson, 1995, p58-59

17

## Examples

- **Designation:**
  - $\text{Mother}(x, m)$  denotes that  $m$  is the genetic mother of  $x$
- **Definition:**
  - $\text{Child}(x, y)$  is defined as  $\text{mother}(y, x)$  or  $\text{father}(y, x)$
- **Refutable Description:**
  - For all  $m$  and  $x$ ,  $\text{Mother}(x, m)$  implies  $\text{not}(\text{Mother}(m, x))$
- **A rough sketch**
  - 'Everyone really belongs to just one family'

Source: Adapted from Jackson, 1995, p58-59

18

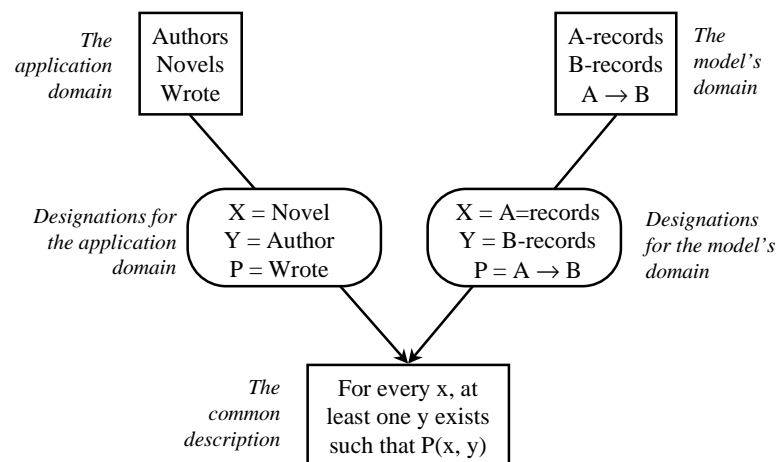
## Models

- **Three types of model**
  - iconic: e.g. a sculpture
  - analogical: e.g. a model airplane
  - analytical: e.g. a set of a mathematical equations representing the economy
- **A model is more than just a description**
  - it has its own phenomena, and its own relationships among those phenomena.
  - The model is only useful if the model's phenomena correspond in a systematic way to the phenomena of the domain being modeled.

Source: Adapted from Jackson, 1995, p120-122

19

## Modeling Example



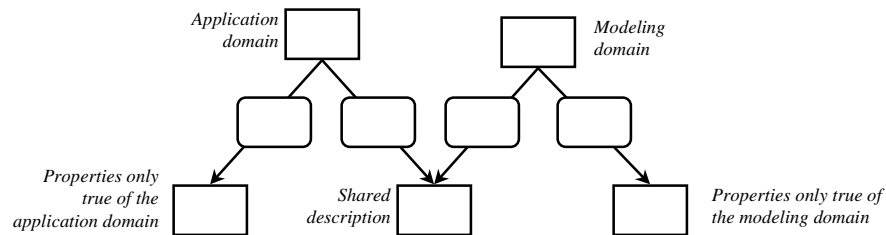
Source: Adapted from Jackson, 1995, p123

20

## Dangers of modeling

- **The model is never perfect:**

- There may be phenomena in the model that are not present in the application domain
- There may be phenomena in the application domain that are not in the model



Source: Adapted from Jackson, 1995, p124-5

21

## Classifying Application Domains

- **Difficulty of Problem**

- Classify as Hard (HA) and Not Hard (NH)
- **Hard:** never been done, or past solutions have failed
- **Not Hard:** old problems, whose solutions are well known
- **Examples:**
  - HA: Landing a person on Mars
  - NH: Patient Monitoring

- **Relationship in time between data and processing**

- Classify as Static (ST) or Dynamic (DY)
- **Static:** all input data available before processing starts
- **Dynamic:** data continues to arrive during processing (includes all interactive and process control systems)
- **Examples:**
  - ST: Payroll
  - DY: Word Processor

Source: Adapted from Davis, 1990, p30

22

## More axes...

- **Number of simultaneous tasks:**
  - Classify as Sequential (SE) or Parallel (PA)
  - Examples:
    - SE: Compiler
    - PA: Telephone Switching
- **Relative Difficulty of data, control and algorithmic aspects of the problem**
  - Which is hardest to specify: Data (DA), Control (CO) or algorithm (AL)
  - Data-hard: complex data moves across system boundary
  - Control-hard: how does the system control its environment (or vice versa)
  - Algorithm-hard: what processing must the system perform.
  - Many application domains exhibit more than one of these.
  - Examples:
    - DA: Payroll
    - CO: Patient Monitoring
    - AL: Compiler

Source: Adapted from Davis, 1990, p31

23

## More Axes...

- **Deterministic vs. Non-deterministic**
  - (Predictability of output for given input)
  - Deterministic (DE): same answer given the same inputs
  - Non-deterministic (ND): system's responses are not well understood; systems have to make decisions using partial information.
  - Examples:
    - DE: Compiler
    - ND: Disease diagnosis

Source: Adapted from Davis, 1990, p32

24

## Categorization Exercise

- **Using:**
  - HA/NH; ST/DY; SE/PA; DA/CO/AL; DE/ND
- **Classify:**
  - A patient monitoring System that sounds alarms whenever a patients vital signs exceed acceptable ranges
  - An elevator control system that controls the movement of elevators and dispatches them to appropriate floors
  - A robot lawnmower that can be place on any lawn and will cut all contiguous areas of grass without hitting such items as shrubs, sidewalks, and trees.
  - An automatic hair cutter that you sit under after telling it what style of haircut you want and which will cut your hair accordingly
  - A payroll program that accepts time cards and generates correct paychecks
  - A private automatic branch exchange (PABX) that provides telephone services
  - A fully automatic automobile production line

Source: Adapted from Davis, 1990, p35

25

## Systems Engineering

- **Definition**
  - There is no standard understanding of system engineering
  - More of a philosophy than a discipline
  - Challenge: Ensure development of optimum solution to meet all technical requirements and provides proper balance of performance, cost, and schedule
- **System Engineer's Responsibilities:**
  - Technical Interface with customer
  - Requirements definition, management, analysis, and flowdown
  - Verification planning and audit
  - Validation planning and audit
  - Interface Management
  - Risk and Opportunity analysis and management
  - Change management and configuration control

Source: Adapted from Forsberg & Mooz, 1997, p44

26

## Challenges for Systems Engineering

- **User Requirements are often not well identified and documented**
- **Insufficient system studies and analyses performed during study period**
- **Specs contain TBDs that the customer will not commit to resolving by a specified date**
- **System concept and operational environment are not well understood**

*Source: Adapted from Forsberg & Mooz, 1997, p47*

27

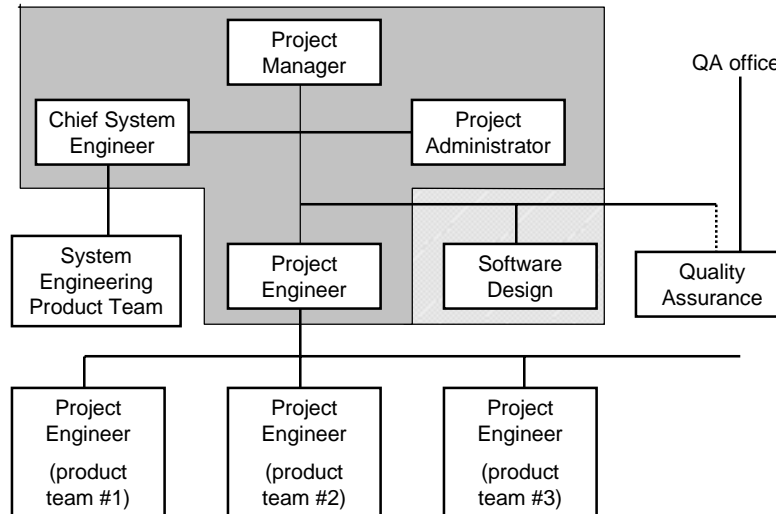
## ...More challenges

- **The problem context and user interface are not well understood**
- **Schedule and budget estimates are unrealistic**
- **Insufficient preparation for system operation**
- **Rapidly changing technology (hence pressure to shorten lifecycle)**
- **Pressure to shorten lifecycle creates pressure to accept point designs**

*Source: Adapted from Forsberg & Mooz, 1997, p48*

28

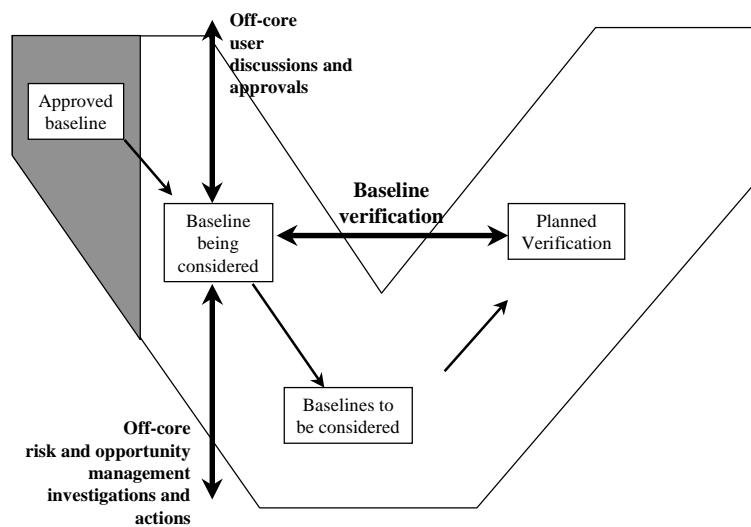
## Typical Org structure



Source: Adapted from Forsberg & Mooz, 1997, p52

29

## V-model for Systems Engineering



Source: Adapted from Forsberg & Mooz, 1997, p55

30

## Next Week

- **Software Requirements Specifications**
  - **Qualities of a good SRS**
  - **Documentation Standards**
- **Formal Inspection exercise**

## References

- Jackson, M. "Software Requirements & Specifications: A Lexicon of Practice, Principles and Prejudices". Addison-Wesley, 1995.
- Davis, A. M. "Software Requirements: Analysis and Specification". Prentice-Hall, 1990.
- Forsberg K. and Mooz H. "System Engineering Overview". In Thayer, R. H and Dorfman, M. (eds.) "Software Requirements Engineering, Second Edition". IEEE Computer Society Press, 1997, p44-72